marconi instruments

The GPIB Manual

Instruction Manual
H 54811-010P

The GPIB Manual

A guide to the MI General Purpose Interface Bus

©
1979

MARCONI INSTRUMENTS LIMITED
ST. ALBANS HERTFORDSHIRE ENGLAND

CONTENTS

PRELIMINARIES

Title page
Contents

CHAPTERS

1 General information
2 Interface functions, subsets and messages
3 System assembly
4 Message format for instrumentation
5 Language description
6 Common programming sequences

Chapter 1

GENERAL INFORMATION

CONTENTS

Para.
1 Scope and objectives of this manual
2 Terminology
3 Background to the GPIB
7 Overview of the main GPIB features

Fig. Page
1 GPIB signal lines and messages 4

SCOPE AND OBJECTIVES OF THIS MANUAL

1. The General Purpose Interface Bus (GPIB) is a standard for the interconnection of programmable instruments to their controlling source and provides a wide range of functional capability for the transfer of information between instruments in many types of system. The Standard was first defined in the American National Standard: "IEEE 488 (1978) Digital Interface for Programmable Instrumentation" and since then the IEC has developed an equivalent international standard defined in Publication 625-1: "An Interface System for Programmable Measuring Apparatus". This differs in two respects, namely a different mechanical specification and an additional section covering recommendations on message format. It is anticipated that a new revision of the American document will also include the additional material. The purpose of this manual is to describe the principles of operation of GPIB Systems and the range of optional facilities, with particular reference to Marconi Instruments OA 1800 series GPIB Controller, to enable users to construct and program systems.

TERMINOLOGY

2. The following specialist terms are used throughout this manual with the particular meaning indicated below although some may have more general meaning in other contexts.

ATE	Automatic test equipment.
GPIB	General Purpose Interface Bus i.e. this interconnection standard.
Bus	A set of parallel wires directly connected to all parts of a system. Normally short for the GPIB itself.

System	A set of interconnected elements constituted to achieve a given objective - normally an ATE.
Device	Any programmable instrument or other apparatus connected to the System that communicates via the GPIB.
Interface	That part of a device which accepts the bus at one port and communicates with the instrument proper at the other.
Programmable	That characteristic of a device that makes it capable of accepting data to alter the state of its internal circuits to perform a specific task.
Signal	The physical representation of information as a digital electrical quantity.
Signal line	One of a set of signal conductors in an interface system used to transfer messages among interconnected devices.
Message	A quantity of information transferred between two distinct parts of a system.
Interface message	A message over the bus used to manage the interface system itself.
Device dependent message	A message over the bus used to control or inform connected devices, which are carried but not used by the interface.
Local message	A message from an instrument proper to its own bus interface.
Bit	Binary digit.
Bit parallel	A set of concurrent data bits present on a like number of signal lines.
Byte	A group of 8 bits operated on as a group.
Address	An 8 bit pattern, unique within a system which is assigned to each device by manual switches within it, which is used to indicate when the device must take part in message transfer over the bus.
Byte serial	A sequence of data bytes carried on 8 parallel data lines.
Status byte	A special data byte which indicates the overall condition of a device.
Handshake	An interlocked sequence of control signals whereby the transfer of one byte of information may be accomplished between devices of differing response times.

Talker	A device with the ability to send device dependent messages over the bus.
Listener	A device with the ability to receive device dependent messages over the bus.
Controller	A device with the ability to send interface messages over the bus. This is a strict definition relating to a specific GPIB function. Within ATE the term is more generally used to mean a unit designed to control groups of instruments and to provide the operator interface. For systems built using GPIB such a unit would certainly contain the bus controller function.

BACKGROUND TO THE GPIB

3. Programmable instruments have in the past been controlled by parallel wire, separate wire per function types of interface with a wide variety of electrical and code specifications. With this type of system the number of wires vary enormously between instruments, for example a synthesiser may have 80 control wires and 1 output, a digital voltmeter could have 6 control and 30 output wires, and a simple transducer may just require 3 controls. It was the responsibility of the ATE manufacturer to provide the necessary interfacing electronics between such instruments and the controlling source.

4. In contrast the GPIB is an interface standard calling for a common bi-directional bus between all devices in a system. The bus has a fixed small number of signal lines (16) which carry all control and measurement information, in a byte serial, bit parallel format, usually coded into an ISO-7 (or ASC11) character string. Thus with the new system architecture all instruments require a bus interface either attached to or built as part of them, and no separate system control electronics are normally needed.

5. The standards referred to above define the electrical, mechanical and functional performance of the interface bus, but allow a range of optional facilities to make the bus suitable for a wide range of applications connecting instruments with an equally wide range of performance. Instruments conforming to the standard can always be safely connected together, but those having different subsets of the options available may not operate functionally together or may require special procedures in the system software to overcome incompatibilities. This is particularly true of device dependent message format, the standards for which are still being developed.

6. The GPIB provides a means of interconnecting up to 15 instruments into a system which is limited to a total bus length of 20 metres and a data rate of 1 megabit per second on any signal line. It has been designed to enable the direct connection of independently manufactured instruments which may have widely differing complexities so that they may communicate asynchronously over a wide range of data rates. A system can be easily modified or extended by plugging the bus cables together in daisy chain fashion.

OVERVIEW OF THE MAIN GPIB FEATURES

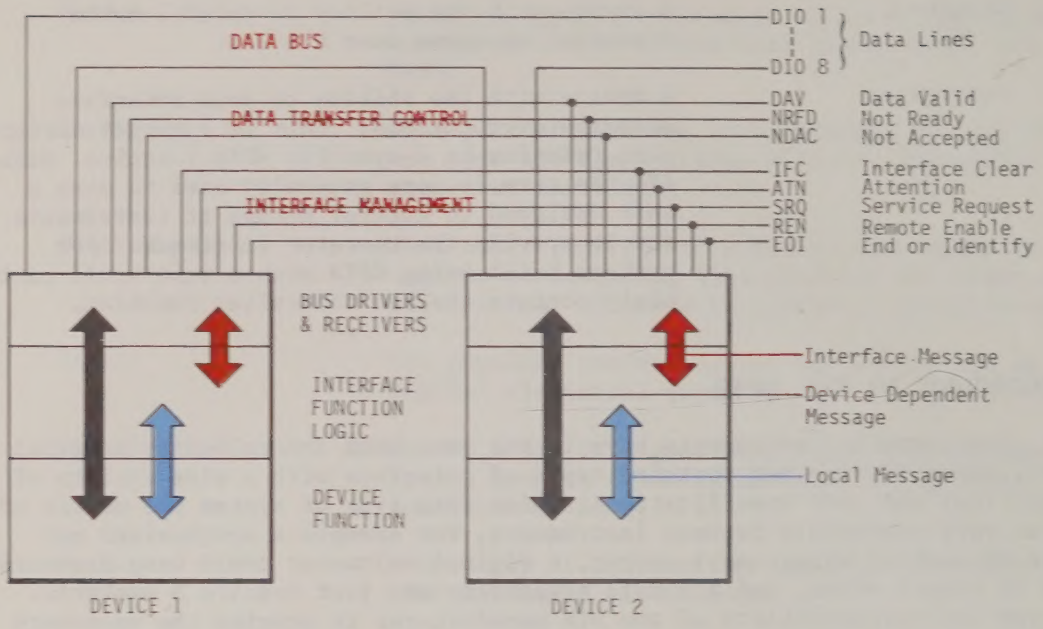


Fig. 1 GPIB signal lines and messages

7. The bus contains sixteen signal lines which are divided into three groups as shown in Fig. 1. The data bus of eight lines is used for all device dependent messages and a high proportion of interface messages which take the form of a single byte. The true (or logical '1') state of the data lines (and all bus lines) is low voltage, and the false (logical '0') is high voltage, both being approximately TTL standard levels. The transfer of each byte of information over the data bus is controlled by a three-wire handshake process between the source of the byte and all the destination device interfaces, using the DAV, NRFD, NDAC lines. The handshake cycle, which is described in detail in Chap. 2, guarantees correct information transfer with any combination of device response times. The third group of signals are the interface management lines, the purpose of which is as follows:

IFC — Interface Clear - used by a controller to place the whole of the interface system into a known quiescent state.

ATN — Attention - specifies in general whether the data bus holds interface messages (true) or device dependent messages (false).

SRQ — Service Request - used by a device to request an interruption of the current bus traffic.

REN — Remote Enable - used by a controller to enable and disable device interfaces.

EOI — End or Identify - is a dual purpose line used either to indicate the end of a multibyte message or as a strobe in a parallel poll sequence.

8. Fig. 1 also shows three basic divisions within each device. The bus transmitters and receivers perform no logical functions on any signal lines and will not therefore be described here, except to point out that more than one type of driver may be used and the choice affects the possible data rate over the bus. The second section is the interface function logic which performs the following tasks:-

Controls the handshake cycle.

Responds to interface messages.

Responds to its own device address.

Transmits to or from the device functions appropriate device dependent messages.

Reacts to local messages from the device function.

9. The device function is the programmable instrument proper, and in the case of traditionally designed instruments the boundary between it and the interface functions is equivalent to the parallel wire programming interface.

10. Interface messages take two forms, namely single bus signal line transitions as described above and coded bytes on the data bus. For the latter the code space is divided into five parts which are separateable by decoding data lines 5, 6 and 7. The types of interface message assigned to each part are as follows:-

Octal data	*Use*	*ISO-7 code representation*
0 - 17	Addressed Commands	Non printing control characters
20 - 37	Universal Commands	
40 - 77	Listener Addresses	Numerals and punctuation
100 - 137	Talker Addresses	Upper case letters
140 - 177	Secondary Addresses	Lower case letters

Data line 8 is reserved for use as a parity bit if required. Detailed assignment of the code space is given in Chap. 2 para. 36.

11. The GPIB provides a range of fundamental communication capabilities which may be implemented as required by the characteristics of a particular device. Within each function listed below a number of options may exist which are detailed in Chap. 2, only the essential purpose of each is described here:

Source Handshake Acceptor Handshake	Functions which manage information transfer over the bus but provide no specific capability and are invisible to the user.
Listener	All device dependent data is received by the listener function when the device's own address has been sent by the controller. A device may be addressed by either a one or a two byte sequence. Virtually all instruments require this function to receive data for their programmable parameters e.g. range, level, etc.

Talker	Any instrument that can generate output data requires the talker function, the purpose of which is to transmit when requested by the controller either measurement data, program data or (as part of a serial poll) the status byte.
Service Request	This function enables a talker to indicate to the controller that it has some information that it wishes to transmit. This may be a measurement which is now complete or perhaps the presence of some error condition. On receipt of the service request message, the controller will normally obtain the status bytes of all possible devices in sequence to ascertain which requested service. This sequence is known as Serial Poll.
Parallel Poll	The parallel poll function provides a device with the ability to transmit on the controllers command one bit of status information simultaneously with several other devices.
Controller	A device with controller function can address other devices to listen or talk and in addition send interface messages to all, or just addressed, devices. In general, the controller manages the bus traffic and it is frequently the case that a device with controller function controls the system in a more general sense, and for that purpose also contains the talker function to transmit a programmed sequence of control messages to instruments in the system (device dependent messages).
Device Clear Device Trigger	These functions provide a device with the ability to be initialised or triggered on command from the controller. This may occur simultaneously with other selected or all devices in a system.
Remote/Local	This function provides a device with the ability to select between program data from the bus and local data, from front panel controls for example.

12. With the exceptions of two byte addressing, all of these functions may be used with the OA 1800 Controller. The bus connector on the OA 1800 Controller conforms to the American Standard, but instruments using the IEC Standard Connector may be coupled into the system using a simple adapter.

13. Although the standard allows data rates up to 1 Mbit/s on a signal line this is only achieved under ideal conditions with limited bus length and loading, and interfaces with response times faster than the minimum permitted. In order to give maximum compatibility between instruments and for high reliability of operation, the OA 1800 Controller is designed for systems where the data rate does not exceed 100 kbit/s.

Chapter 2

INTERFACE FUNCTIONS, SUBSETS AND MESSAGES

CONTENTS

Para.

1 Introduction
3 Source handshake function
3 Definition
4 Options
5 Sequence
6 Acceptor handshake function
6 Definition
7 Options
8 Sequence
9 Combined source and acceptor handshake timing
11 Talker function
11 Definition
12 Options
13 Sequence
15 Listener function
15 Definition
16 Options
17 Sequence
18 Service Request function
18 Definition
19 Options
20 Sequence
21 Remote Local function
21 Definition
22 Options
23 Sequence
25 Device Clear function
25 Definition
26 Options
27 Sequence
28 Device Trigger function
28 Definition
28 Options
29 Sequence
30 Controller function
30 Definition
31 Options
32 Sequence
36 Summary of interface messages

Table		Page
1	Talker function options	8
2	Controller function options	18
3	Single line interface messages	21
4	Universal commands	21
5	Addressed commands	22
6	Addresses	22

Fig.		Page
1	Source Handshake state diagram	3
2	Acceptor Handshake state diagram	5
3	Correlation of handshake signals and states	6
4	Talker state diagram	9
5	Listener state diagram	11
6	Service Request state diagram	12
7	Remote/Local state diagram	14
8	Device Clear state diagram	15
9	Device Trigger state diagram	16
10	Controller state diagram	19

INTRODUCTION

1. In the published standards the interface functions are precisely defined using state diagrams with mnemonics for all states and Boolean expressions for all transitions. These are accompanied by verbal descriptions optimised for the interface circuit designer. That level of detail is unnecessary and maybe confusing to the system user, but it is important to understand the practical implications of the operational sequences within each function. In this chapter the basic form of the state diagrams have been reproduced but with only verbal descriptions of the practical signals and actions, but they correlate directly to those of the published standard.

2. In each state diagram the each circle represents a different state in which the interface maybe. Some may be stable indefinitely, others for a pre-determined time only. The arrows show the transitions that are possible between states and the conditions under which each transition must be made. If a transition is caused by a local condition (message) then it is represented in blue, whereas interface messages from the bus cause transitions in red.

SOURCE HANDSHAKE FUNCTION : SH

Definition

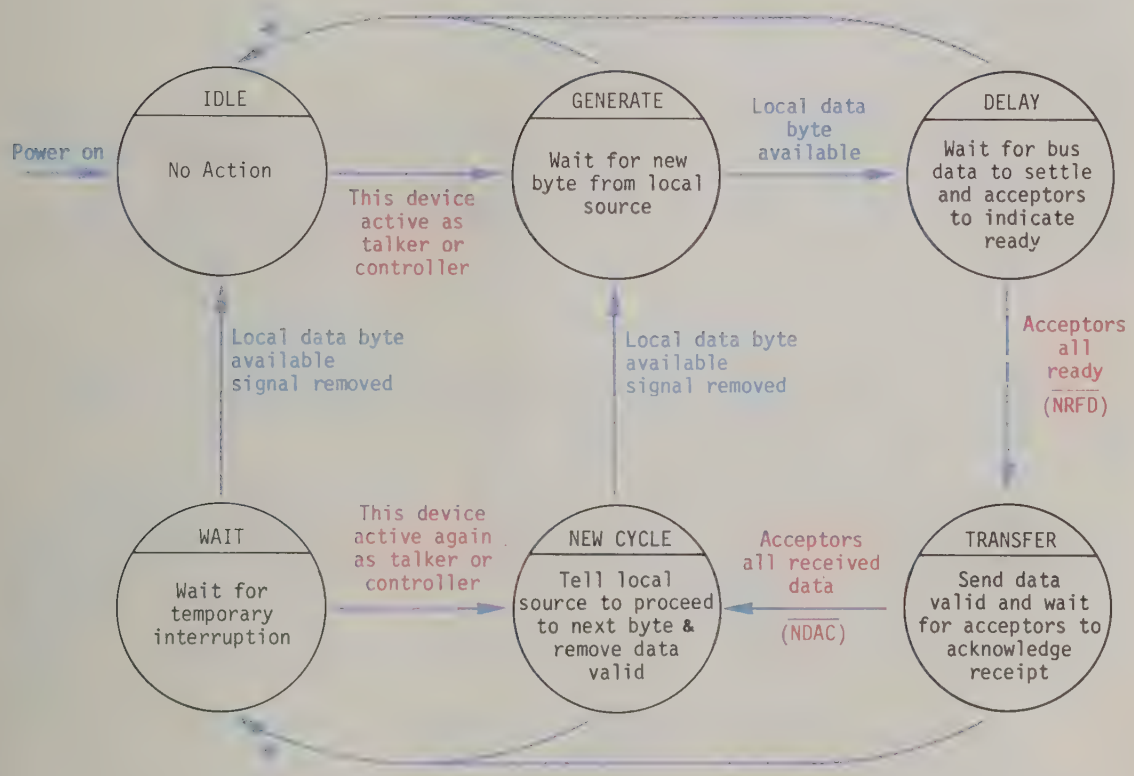
3. The source handshake sequences the transmission of each data byte over the bus data lines. The content of that byte may be an interface message from a controller or (part of) a device dependent message from a talker. The sequence is initiated when the controller or talker becomes active, and the purpose of the function is to synchronise the rate at which bytes become available to the rate at which accepting devices on the bus can receive the data.

Options

4. Apart from being present or not, there are no options. Only a few of the simplest instruments which have no data output and cannot report their status omit this function.

IDENTIFICATION	*DESCRIPTION*	*STATES OMITTED*	*OTHER FUNCTIONS REQUIRED*
SH 0	No capability	All	None
SH 1	Complete capability	None	T1-T8 or C5-C28

Sequence


IDLE
No Action
Power on
This device active as talker or controller
GENERATE
Wait for new byte from local source
Local data byte available
DELAY
Wait for bus data to settle and acceptors to indicate ready
Local data byte available signal removed
Local data byte available signal removed
Acceptors all ready
(NRFD)
WAIT
Wait for temporary interruption
This device active again as talker or controller
NEW CYCLE
Tell local source to proceed to next byte & remove data valid
Acceptors all received data
(NDAC)
TRANSFER
Send data valid and wait for acceptors to acknowledge receipt

* This device's activity as talker or controller is terminated

Note: Transitions between states are triggered by interface or local messages

Fig. 1 Source Handshake state diagram

5. Fig. 1 shows the source handshake state diagram. At the start, i.e. power on, it is in the idle state. The following signals and messages are required to effect the appropriate state transitions.

Bus signal lines : NRFD Not ready for data

NDAC Not data accepted

and ATN Attention

in combination with the active state from either the talker or controller function.

Local message "new data byte available".

The bus data lines must be updated in the GENERATE state and held valid by the source handshake in both DELAY and TRANSFER states.

ACCEPTOR HANDSHAKE FUNCTION : AH

Definition

6. The acceptor handshake sequences the reading of each data byte from the bus data lines and ensures error free reception by transmitting the NRFD and NDAC signals at times appropriate to its own speed of response. The data byte received may be an interface message from the controller or (part of) a device dependent message from a talker, and must be routed within the device accordingly.

Options

7. Apart from being present or not, there are no options. Only in the unlikely event of an instruments' capability being limited to talking continuously would this function be omitted.

IDENTIFICATION	*DESCRIPTION*	*STATES OMITTED*	*OTHER FUNCTIONS REQUIRED*
AH 0	No capability	All	None
AH 1	Complete capability	None	None

Sequence

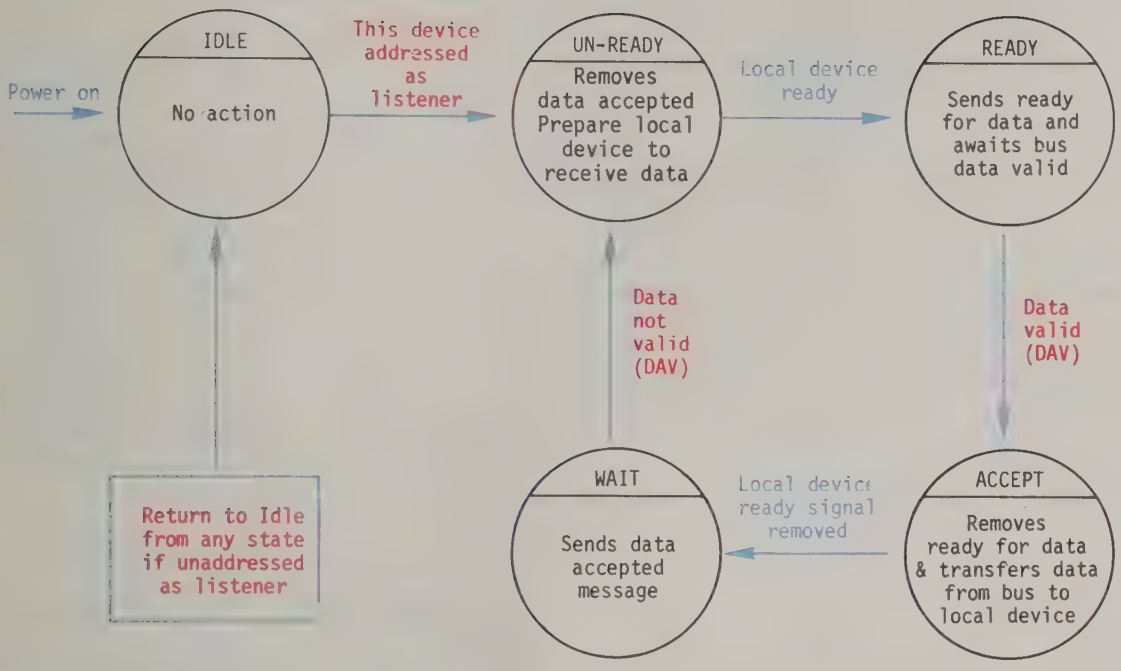

IDLE
No action
Power on
This device addressed as listener
UN-READY
Removes data accepted Prepare local device to receive data
Local device ready
READY
Sends ready for data and awaits bus data valid
Data not valid (DAV)
Data valid (DAV)
Return to Idle from any state if unaddressed as listener
WAIT
Sends data accepted message
Local device ready signal removed
ACCEPT
Removes ready for data & transfers data from bus to local device

Fig. 2 Acceptor Handshake state diagram

8. The state diagram, Fig. 2, is shown for accepting device dependent message bytes. For receiving interface messages the transitions from and back to the idle state are demanded by the ATN bus signal line being asserted or removed, and in this case the transition from UNREADY to READY is instantaneous and that from ACCEPT to WAIT is in a fixed time.

Signals received are:

Bus signal lines :	DAV	Data valid
	ATN	Attention
Other function states :	Listener addressed or Listener active	
Local message :	Ready to accept new data byte.	

The full acceptor handshake diagram also relates to the passing of control in multi-controller system. For clarity this is omitted here, and the reader should refer to the full standard for this operation.

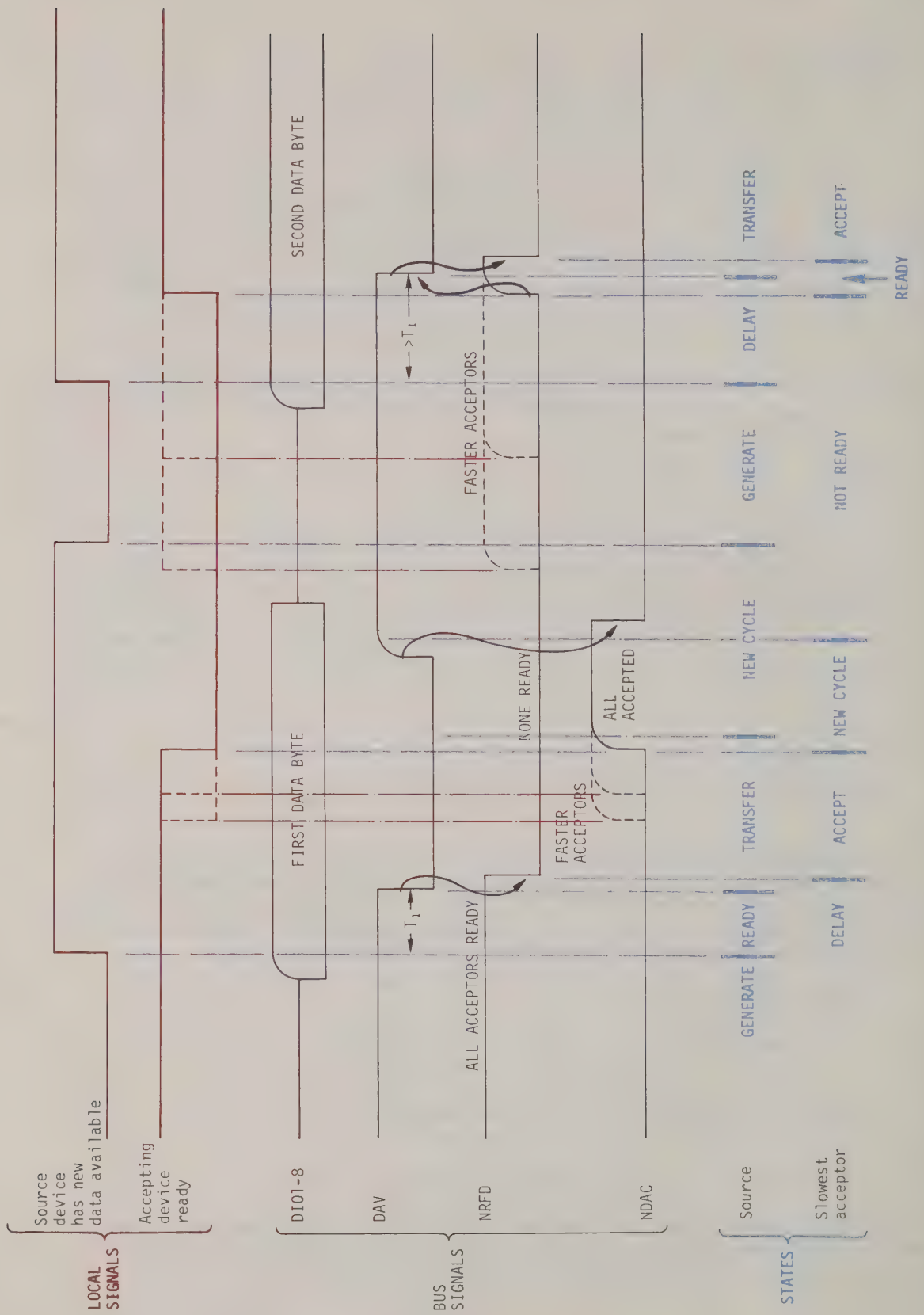


Fig. 3 Correlation of handshake signals and states

Combined Source and Acceptor Handshake timing

9. Fig. 3 shows the practical timing signals of the combined three-wire handshake. The NRFD and NDAC lines are shared by all listeners and hence the signals received on these lines by the talker is the logical OR combination of those sent by the individual listeners. The voltage waveform seen on the lines will obviously be that of the slowest acceptor in each case, and the dotted transitions in the diagram represent the faster acceptors reacting to their device functions internal response time.

10. The sequence starts assuming all acceptors are ready and the source is generating the first data byte. The data is placed on the bus as soon as it is available and a minimum time of T1 is allowed in the delay state before DAV is asserted. The slowest acceptor releases NDAC when its device function has received the data and when this is received the source begins the generation of a new byte and releases the DAV signal. The speed with which acceptors can become ready for the next byte is variable, and in this example one acceptor is not ready until after the second data byte has been transmitted, and therefore the new DAV signal follows immediately NRFD goes high. In this way devices of all responses times are synchronized, and the delay caused by the slowest device is minimized.

TALKER FUNCTION : T

Definition

11. The talker function provides a device with the ability to send device dependent messages over the bus to other devices. Any instrument which generates data (e.g. measurement results) or which needs to transmit status information to the controller, therefore needs the talker function which includes serial poll response sequence. The standard has two versions of talker function, for one or two address byte systems. The OA 1800 Controller only uses the former, which is described below. The same is also true of the listener function. For details of the two byte addressing systems refer to the Extended Talker (and Extended Listener) functions in the standard. The ability of a device to talk only exists when it has been addressed as a talker.

Options

12. There are three basic variables in the design of a single-byte addressable talker, and combinations of these give rise to eight possible subsets of the function, plus TO (no capability). The options are:

a) Ability to response to serial poll by placing the device's status byte on the bus. The coding of the status byte is covered in Chap. 4, but it must contain service request in bit seven and it normally contains busy/ready and error information on other data lines.

b) The device may have a manual switch to permanently address the talk function. This is referred to as talk only mode and is useful when a device happens to be the only talker in a system.

c) The talker function may be automatically unaddressed if the device is addressed as a listener.

TABLE 1 TALKER FUNCTION OPTIONS

IDENTIFICATION	*DESCRIPTION*			*STATES OMITTED*	*OTHER FUNCTIONS REQUIRED*
T0	No capability			All	None
	Serial Poll	Talk only	Unaddress on Listen		
T1	Y	Y		None	SH1 & AH1
T2	Y			None	SH1 & AH1
T3		Y		POLL IDLE,MODE & ACTIVE	SH1 & AH1
T4				POLL IDLE,MODE & ACTIVE	SH1 & AH1
T5	Y	Y	Y	None	SH1 & L1-4 (& AH1)
T6	Y		Y	None	SH1 & L1-4 (& AH1)
T7		Y	Y	POLL IDLE,MODE & ACTIVE	SH1 & L1-4 (& AH1)
T8			Y	POLL IDLE,MODE & ACTIVE	SH1 & L1-4 (& AH1)

Sequence

13. The state diagram can be considered in two parts, firstly omitting the serial poll option, so that three states remain. At power on, or after an IFC (interface clear) bus signal, the function is in the idle state. Interface messages will be being received by the acceptor handshake function, and when it receives the device talk address on the data lines (in the ACCEPT state) then the talk function moves to its ADDRESSED state. Up to this point the controller has been using the bus for interface messages and the ATN (attention) signal is therefore present. The removal of ATN by the controller is the signal that it is relinquishing use of the data bus and DAV to the addressed talker. When the talker sees ATN removed it therefore moves to its ACTIVE state and its source handshake function can start to transfer device dependant message bytes over the bus. The talker only exits the ACTIVE state when the controller interrupts with either the ATN or IFC signals, and there are several criteria that may be programmed into the controller for taking these actions. For example it may see that the talker has finished its message, or another device may have requested service. Since only one device can talk at once, any talk address forces all other talkers back to the IDLE state. If the talk only option is in operation, transitions to the IDLE state are only mementary and the device normally rests in the ADDRESSED state.

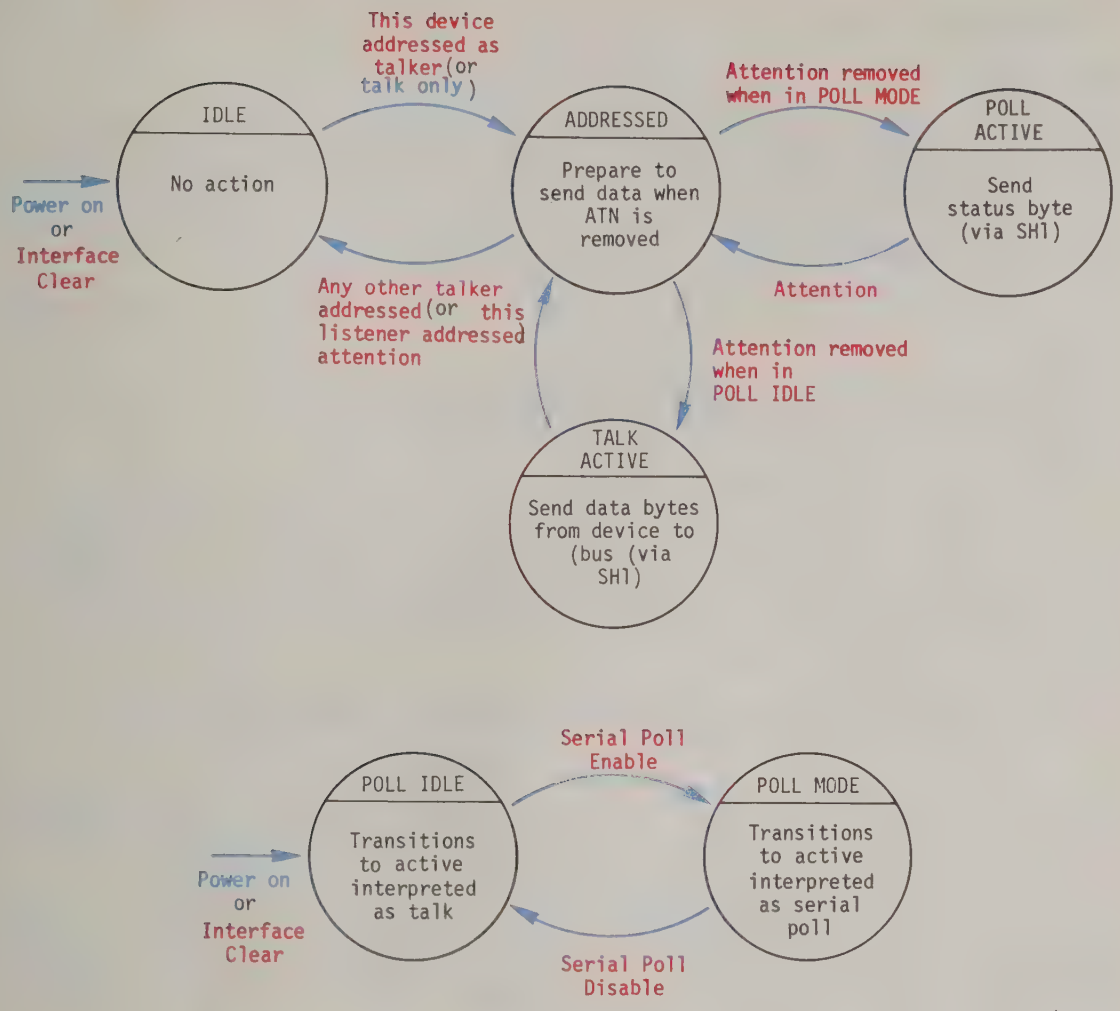

This device addressed as talker (or talk only)
Attention removed when in POLL MODE
IDLE
No action
ADDRESSED
Prepare to send data when ATN is removed
POLL ACTIVE
Send status byte (via SH1)
Power on or Interface Clear
Any other talker addressed (or this listener addressed attention
Attention
Attention removed when in POLL IDLE
TALK ACTIVE
Send data bytes from device to (bus (via SH1)
Serial Poll Enable
POLL IDLE
Transitions to active interpreted as talk
POLL MODE
Transitions to active interpreted as serial poll
Power on or Interface Clear
Serial Poll Disable

Fig. 4 Talker state diagram

14. When the Serial Poll option is present there are two active states which are reached from the ADDRESSED state when ATN is removed. Which is chosen depends on whether the Serial Poll enable message has previously been received and hence which of the two states in the lower diagram is active. Serial Poll Enable and Serial Poll Disable are universal messages sent by the controller over the data lines to all devices simultaneously. In the poll active state the device transmits its status byte over the bus. Normally the controller will re-assert ATN after the single byte, but multiple transfers are possible and in this case the meaning (coding) of the individual bits of the status byte must remain the same, although their value may change. For example if bit 5 is used in the status byte for busy = 1, finished = 0 then it cannot be subsequently used for anything else, but if a string of status bytes were requested, the first few may have bit 5 = 1 and later it can change to 0 indicating when the programmed task is complete.

LISTENER FUNCTION : L

Definition

15. The listener function provides a device with the ability to receive device dependent messages over the bus. The capability only exists when the device is addressed to listen via the bus by the controller. The controller itself has to listen to the responses to serial poll, and in this case a local message to listen is sufficient. Every practical device has the listener function.

Options

16. There are two basic variables in the design of a single byte addressable listener, and these give rise to four optional subsets.

a) The device may have a manual switch to permanently address the listen function. This is known as the listen only mode and is used in very simple systems, usually without a controller. For example a source of measurements may be connected via the bus to a recording device in a continuous monitoring application which is manually set up with the measuring instrument in talk only mode and the recorder in listen only mode.

b) The listener function may be automatically unaddressed if the device is addressed as a talker.

IDENTIFICATION	*DESCRIPTION*		*STATES OMITTED*	*OTHER FUNCTIONS REQUIRED*
LO	No capability		All	None
	Listen only	Unaddress on talk		
L1	Y		None	AH1
L2			None	AH1
L3	Y	Y	None	AH1 and T1-8
L4		Y	None	AH1 and T1-8

Sequence

17. The normal sequence of events for the listener function is as follows. After power on or an IFC (interface clear) message the function is in the IDLE state until the devices own listen address is received over the bus, or the local listen only switch is on. The transition to ADDRESSED state is then made and the device waits for ATN (attention) signal to be removed, indicating that use of the data lines has been handed over to the talker, hence the listener transfers to ACTIVE state in which it receives one device dependent data byte each time its acceptor handshake is in the ACCEPT state. Transitions back to the IDLE state are similar to that of the talker function except that as multiple concurrent listeners are permitted there is a specific un-listen command instead of un-addressing on receipt of any other devices listen address. When in the ADDRESSED state the listener must implement any addressed interface commands.

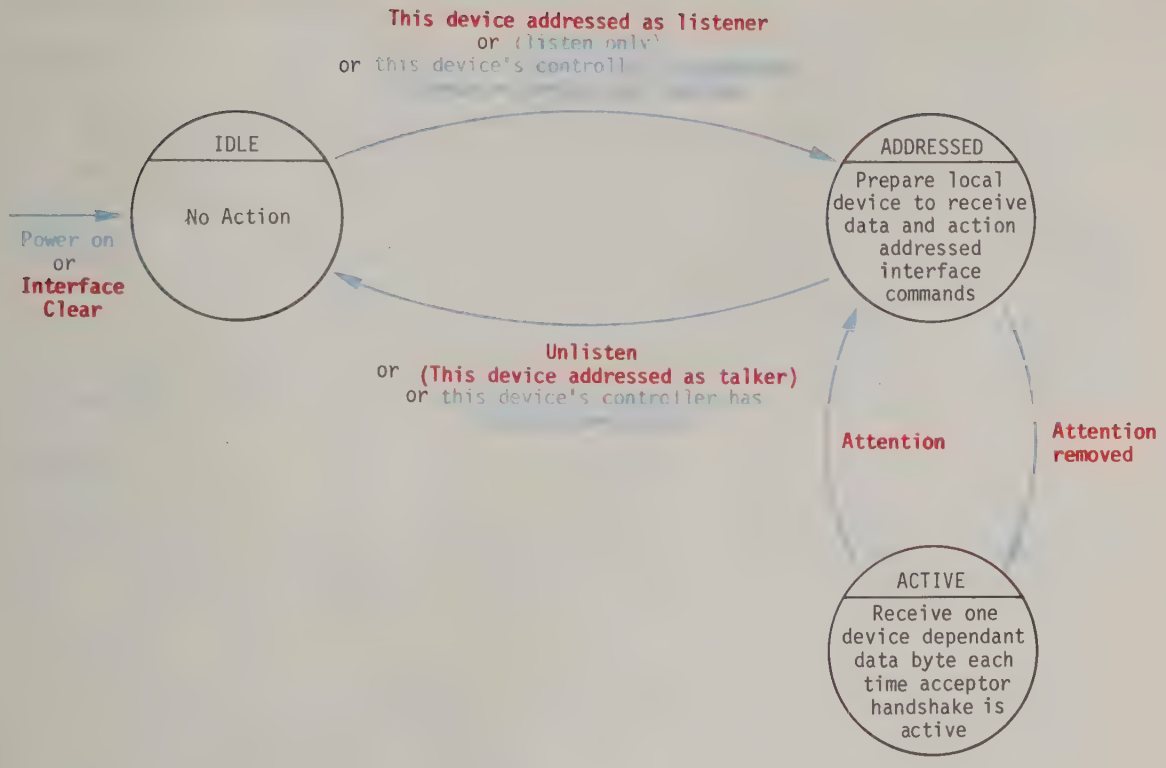


Fig. 5 Listener state diagram

When a controller is conducting serial poll its listen function is used to receive the status information from polled devices. In this case the transitions between IDLE and ADDRESSED state are usually commanded by local messages within the controller, as this is quicker than addressing itself over the bus.

SERVICE REQUEST FUNCTION : SR

Definition

18. The service request function gives a device the capability to asynchronously inform the controller of its need of attention (in a general sense). Most talkers require this to be able to indicate availability of new device dependent information while other bus traffic is in progress. Other uses of the function are to indicate fault conditions e.g. 'I have blown a fuse', or 'I did not understand the last instruction'. Alternatively, it may report manual interactions such as 'I have been switched to local control'. Within one device interface the function must be duplicated for each reason for the service request, although all will signal on the SRQ signal line and all will set bit 7 in the status byte.

Options

19. Apart from being implemented one or more times or not at all, there are no options.

IDENTIFICATION	*DESCRIPTION*	*STATES OMITTED*	*OTHER FUNCTIONS REQUIRED*
SR 0	No capability	All	None
SR 1	Complete capability	None	T1,2,5 or 6

Sequence

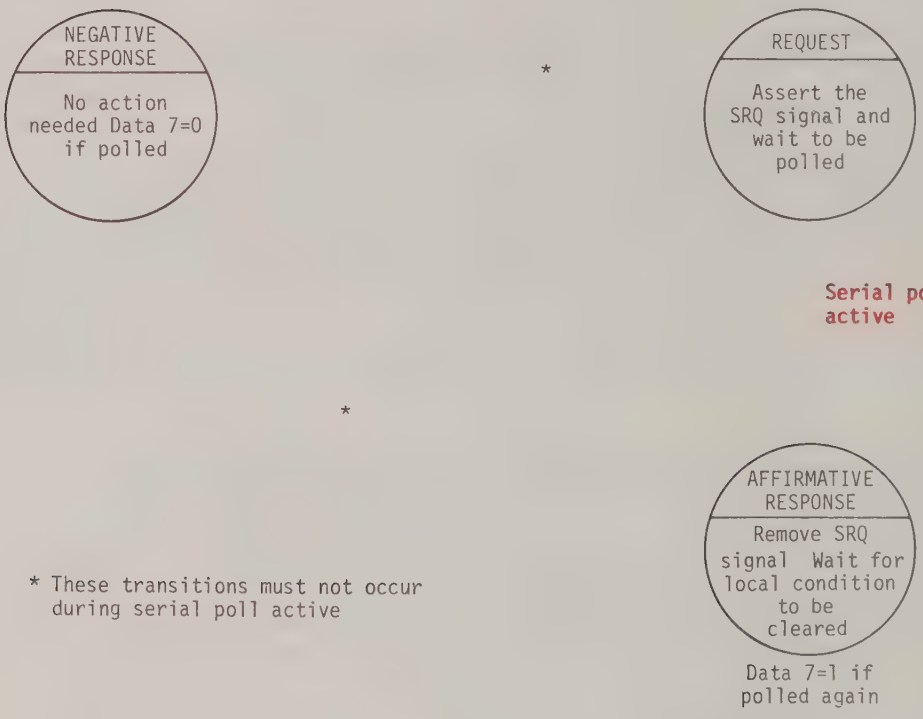

NEGATIVE RESPONSE
No action needed Data 7=0 if polled
*
REQUEST
Assert the SRQ signal and wait to be polled
Serial poll active
*
AFFIRMATIVE RESPONSE
Remove SRQ signal Wait for local condition to be cleared
Data 7=1 if polled again
* These transitions must not occur during serial poll active

Fig. 6 Service Request state diagram

20. The function powers on and waits in the NEGATIVE RESPONSE state until the local device indicates its need for service, and this causes the transition to the REQUEST state. It is important that this transition, and those back to NEGATIVE RESPONSE, do not occur while the devices talker function is in POLL ACTIVE state, which would be equivalent to changing ones mind while being asked if one requested service. It is of course possible that the reason for requesting service disappears without or before a serial poll in which case it is allowable to return immediately to NEGATIVE RESPONSE state. However, in the REQUEST state the SRQ signal line is continuously asserted until a serial poll is conducted and having transmitted the status byte in serial poll ACTIVE state, the Service Request function changes to the AFFIRMATIVE RESPONSE state. In this state SRQ is not asserted, but bit 7 of the status byte continues to be set. The serial poll has ascertained the reason for the Service Request, but only when the controller has taken the appropriate action to correct or process the condition will the bit 7 be cleared, and the function return to NEGATIVE RESPONSE state.

REMOTE/LOCAL FUNCTION : RL

Definition

21. The remote/local function gives a device the ability to switch between two sources of programming information, namely the local front panel controls and device dependent messages over the bus. The function is constructed so that the relative priority of both may be altered allowing either to take precedence under appropriate conditions. It is an option of the designer as to whether the instrument responds only to control messages received after a change from local to remote or whether previously accumulated remote settings are immediately implemented. The remote/local function only affects the use of received control data which has a locally generated equivalent. Being in local control does not stop a device acting as a talker or receiving messages not associated with corresponding local controls.

Options

22. There is one optional part of the remote/local function, which is known as local lockout. This is the ability of the function to receive the universal message Local Lockout and from that time ignore the manual local/remote switch on the device.

IDENTIFICATION	*DESCRIPTION*	*STATES OMITTED*	*OTHER FUNCTIONS REQUIRED*
RL0	No capability	All	None
RL1	Complete capability	None	L1 - L4
RL2	No local lockout	{Local with lockout Remote with lockout}	L1 - L4

Sequence

23. Power on is in the local state and the function returns there if the REN (remote enable) bus signal is removed. In this state the device responds to local controls but must receive remote messages and may store them for future use. In the top two states, the local device has priority and therefore it will remain in local so long as the 'return to local' signal is present. If this local control is removed, the act of addressing the device as listener causes the transition to REMOTE state. The transition back to LOCAL may also be affected by an addressed command over the bus.

24. The lower two states give priority to remote control via the bus, and hence the instruments own remote switch is overridden. In other respects the states with lockout operate the same as those above. The only way to remove local lockout is to remove the REN (remote enable) bus signal. IFC interface clear does not affect the remote/local function.

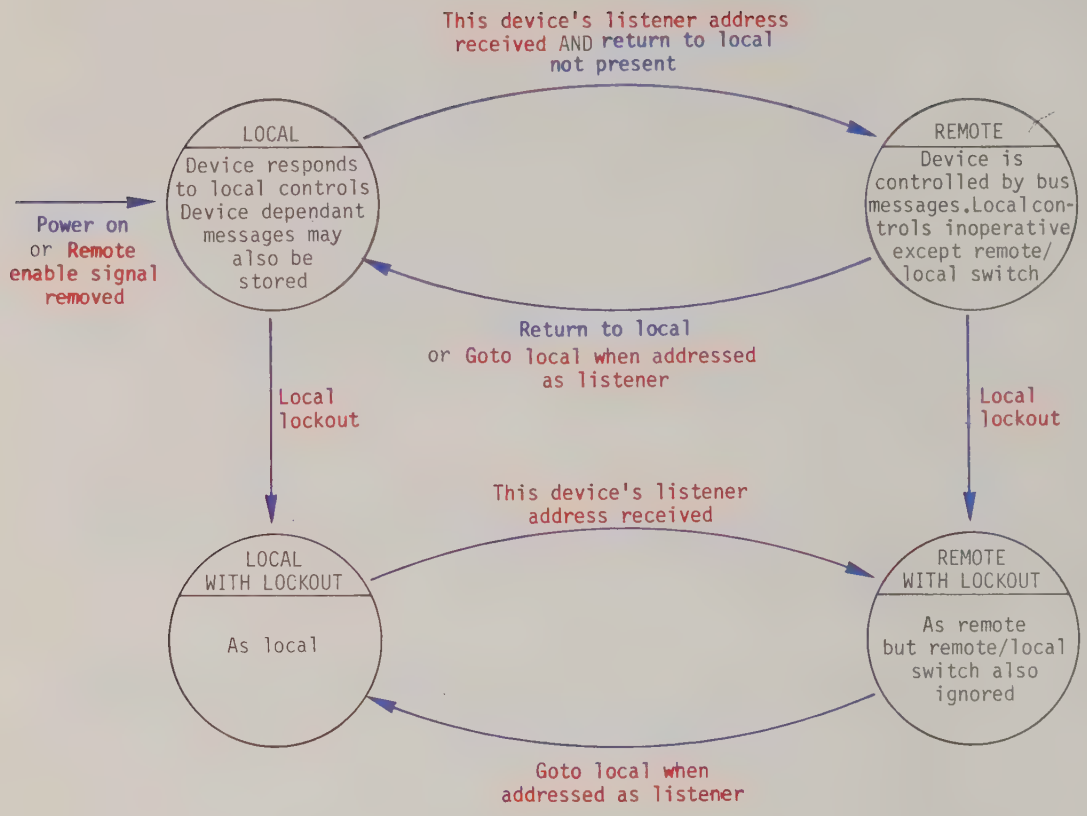

This device's listener address received AND return to local not present
LOCAL
Device responds to local controls Device dependant messages may also be stored
REMOTE
Device is controlled by bus messages. Local controls inoperative except remote/local switch
Power on or Remote enable signal removed
Return to local or Goto local when addressed as listener
Local lockout
Local lockout
This device's listener address received
LOCAL WITH LOCKOUT
As local
REMOTE WITH LOCKOUT
As remote but remote/local switch also ignored
Goto local when addressed as listener

Fig. 7 Remote/Local state diagram

DEVICE CLEAR FUNCTION : DC

Definition

25. Device Clear is a general Reset to an instrument and does the same job as IFC (interface clear) does for the instrument interface. The device clear may be given to all devices in a system simultaneously, or only to addressed devices, providing the function is implemented.

Options

26. The only option within this function is to omit the selective device clear feature. If this is omitted then the same result can still be achieved by addressing the device as listener and sending device dependent messages to clear all functions.

IDENTIFICATION	*DESCRIPTION*	*STATES OMITTED*	*OTHER FUNCTIONS REQD.*
DC0	No capability	All	None
DC1	Complete capability	None	L1 - L4
DC2	No selective clear	None	AH1

Sequence

27. This simple function needs no description except to point out that the ACTIVE state only exists while the acceptor handshake is actually receiving the device clear data byte, a time specific to each device, and normally of the order of a few microseconds.

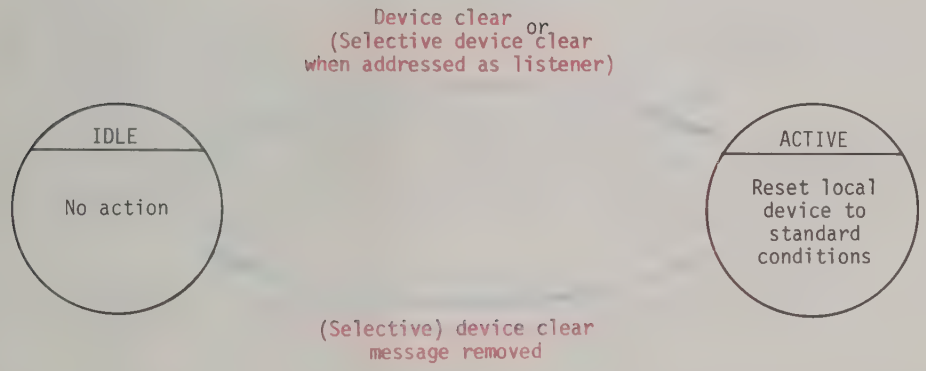


Fig. 8 Device Clear state diagram

DEVICE TRIGGER FUNCTION : DT

Definition

28. Although the bus is designed to allow devices to simultaneously listen to device dependent messages, it is rarely, if ever, the case that control settings are applicable and compatible. The device trigger function is therefore provided to allow the synchronisation of the start of two or more devices tasks irrespective of their particular programming codes.

IDENTIFICATION	*DESCRIPTION*	*STATES OMITTED*	*OTHER FUNCTIONS REQD.*
DT 0	No capability	All	None
DT 1	Complete capability	None	L1 - L4

Sequence

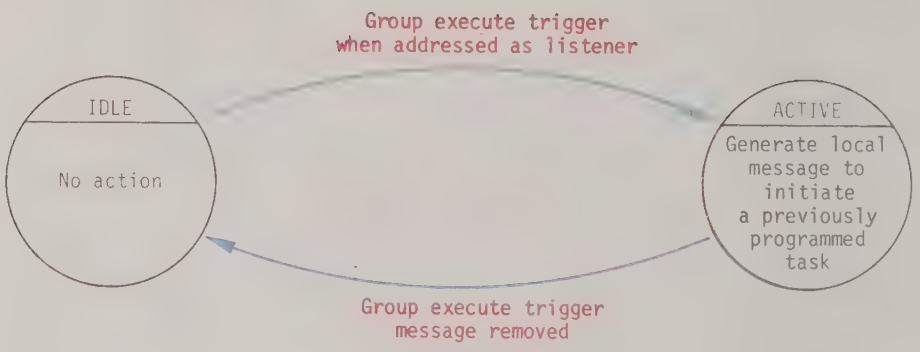


Fig. 9 Device Trigger state diagram

29. This is very similar to the Device Clear function, except that only the addressed command is available.

CONTROLLER FUNCTION : C

Definition

30. The controller function provides a device with the ability to send addresses, universal commands and addressed commands over the bus, and, in general, manages the bus traffic and responds to requests for service from other devices. In systems which have only one device with controller function, its operation is relatively simple, alternating between itself using the data bus for control purposes and permitting use of the data lines by the talkers in the system. In systems with more than one device with controller capability, two additional functional concepts have to be introduced. One (and only one) device must be assigned by manual switch the task of 'system controller'. This device will then assume system control at power on and can demand control of the bus at any later time by means of the interface clear signal over which it has sole control. After the system controller has established control over the bus, it may delegate control of the interface (except IFC and REN) to any other device with controller function. The device

which has bus control at a particular time is known as the 'controller-in-charge'.

Options

31. There are nine options within the controller function which give rise to a large number of possible sub-sets, not all of which are practicable or allowable. Table 2 shows the coding of sub-sets and their dependencies. One or more from the group C1-4 may be present in addition to one from the group C5-28. The basic optional facilities are:

a) Whether it has system control capability and hence can drive the remote enable and interface clear signals.

b) Whether it can exist in a multi-controller system and hence pass or receive charge of the bus.

c) Whether it has the ability to conduct a parallel poll.

d) Whether it has the ability to receive and respond to service requests.

The OA 1800 series Controller has total capability except parallel poll and hence its controller subset is defined as C1,2,3,4,7.

32. If the whole of the controller function is implemented there are five separate parts to the state diagram and the function must be in one state in each part simultaneously whenever the device is powered on. The four smaller parts of the diagram are quite simple and relate to single purposes as follows:

Part 2 indicates whether or not this device has been nominated (by local switch) as the system controller.

Part 3 controls the sending of interface clear signal by the system controller which alternates between the ACTIVE and INACTIVE states. All other controllers do not leave the IDLE state.

Part 4 controls the sending of the remote enable signal in a similar way to Part 3.

Part 5 responds to the SRQ service request signal line. It is then up to the controlling devices logic design what action is appropriate.

The main part of the state diagram has three loop sequences only one of which can be actioned at a time. From IDLE to and from ACTIVE are the required transitions for passing and receiving the controller-in-charge facility. The loop from ACTIVE to STANDBY and WAIT states is used each time a talker uses the data lines to transmit a message. The last group of transitions to the parallel poll states are not implemented in the OA 1800 Controller and will not be further discussed.

TABLE 2 CONTROLLER FUNCTION OPERATIONS

IDENTIFICATION	CAPABILITIES										STATES REQUIRED										OTHER FUNCTION REQUIRED				
	System Controller	Send Interface Clear and Take Charge	Send Remote Enable	Respond to Service Requests	Receive charge of bus	Pass charge of bus	Pass charge to self	Conduct at Parallel Poll	Take control synchronously from Talker		System active and inactive	Interface Clear Idle, Active and Inactive	Remote Enable Idle, Active and Inactive	Service Required and Not Required	Controller Active, Standby, Wait and Wait for EOI	Controller Addressed	Controller Idle	Controller Transfer	Parallel Poll Wait and Active		C1	C2	SH1	AH1, L1-4	T1-8
C0	No capability										None										None				
C1	Y										Y														
C2		Y										Y									Y				
C3			Y										Y								Y				
C4				Y										Y											
C5					Y	Y	Y	Y	Y						Y	Y	Y	Y	Y				Y	Y	Y
C6					Y	Y	Y	Y							Y	Y	Y	Y	Y				Y		Y
C7					Y	Y	Y		Y						Y	Y	Y	Y					Y	Y	Y
C8					Y	Y	Y								Y	Y	Y	Y					Y		Y
C9					Y	Y		Y	Y						Y	Y	Y	Y	Y				Y	Y	Y
C10					Y	Y		Y							Y	Y	Y	Y	Y				Y		Y
C11					Y	Y			Y						Y	Y	Y	Y					Y	Y	Y
C12					Y	Y									Y	Y	Y	Y					Y		Y
C13					Y			Y	Y						Y	Y	Y		Y		Y	Y	Y	Y	
C14					Y			Y							Y	Y	Y		Y		Y	Y	Y		
C15					Y				Y						Y	Y	Y				Y	Y	Y	Y	
C16					Y										Y	Y	Y				Y	Y	Y		
C17						Y	Y	Y	Y						Y		Y	Y	Y		Y	Y	Y	Y	Y
C18						Y	Y	Y							Y		Y	Y	Y		Y	Y	Y		Y
C19						Y	Y		Y						Y		Y	Y			Y	Y	Y	Y	Y
C20						Y	Y								Y		Y	Y			Y	Y	Y		Y
C21						Y		Y	Y						Y		Y	Y	Y		Y	Y	Y	Y	Y
C22						Y		Y							Y		Y	Y	Y		Y	Y	Y		Y
C23						Y			Y						Y		Y	Y			Y	Y	Y	Y	Y
C24						Y									Y		Y	Y			Y	Y	Y		Y
C25								Y	Y						Y				Y				Y	Y	
C26								Y							Y				Y				Y		
C27									Y						Y								Y	Y	
C28															Y								Y		

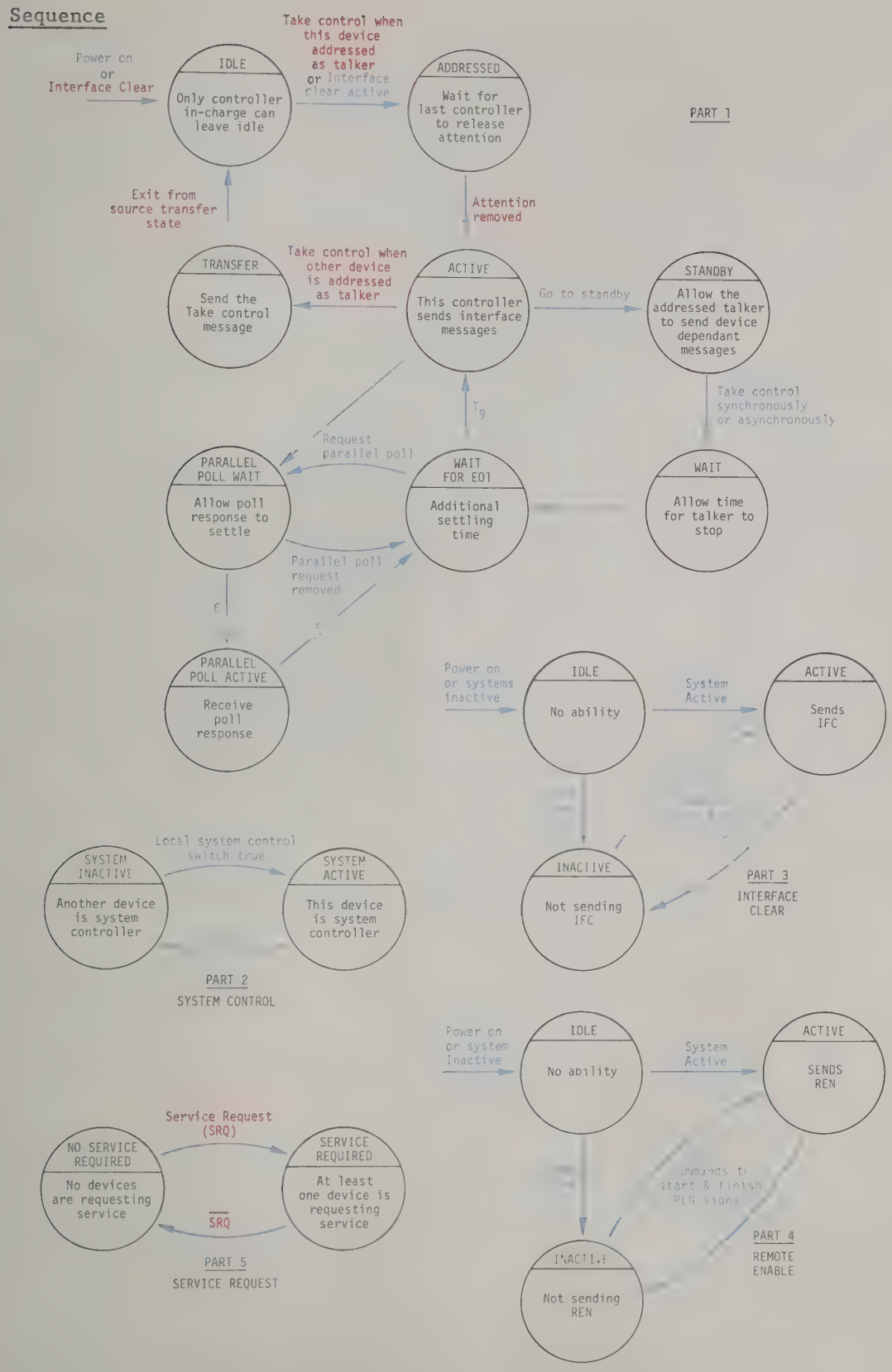


Fig. 10 Controller state diagram

33. It is an important requirement of multi-controller systems that at all times there is no conflict between them, otherwise unambiguous message transfer over the bus could not be achieved. For this reason, all controllers power up in the IDLE state and only that one which has been previously appointed system controller can by itself transfer to the ACTIVE state from which interface messages may be sent. The only way in which another controller may become ACTIVE (i.e. in-charge) is for it to be addressed as a talker by the ACTIVE controller and for the 'take control' addressed command to be transferred between them via the source and acceptor handshakes. Once ACTIVE, any controller may assume its normal tasks of sending interface messages and setting up message paths for device dependent data exchange.

34. The controller-in-charge may only lose control by deliberately passing control on to another device or by the system controller interrupting with the interface clear signal, in which case the current bus activity is aborted for all devices.

35. When the controller-in-charge is in STANDBY state, a talker is assumed to be using the data lines for a device dependent message. It is the responsibility of the controller to decide at which time to take back control of the data bus and there are several criteria which may be programmed into the design of the controlling device. The controller may wait for the end of the talker's message by monitoring either the EOI signal line or the message itself looking for the terminating character. Alternatively, the controller may have received a service request and decide to interrupt the talker to service the other device. In either case, the controller asserts ATN signal but it is preferable to do it when the acceptor handshake is in the NOT READY state, when no corruption of device dependent data is possible. This is known as take control synchronously. The controller may take control asynchronously if the situation is urgent but this should be used with caution since there is no other way of detecting whether the data byte being sent has been correctly received by all listeners.

SUMMARY OF INTERFACE MESSAGES

36. Tables 3 to 6 give detailed assignment of the code space.

TABLE 3 SINGLE LINE INTERFACE MESSAGES

NAME	*SENDER*	*RECEIVER*	*PURPOSE*	*MNEMONIC*	*DATA LINE USAGE*
Attention	Controller-in-Charge	All devices	Indicate Interface message	ATN	Multi-line message
Interface Clear	System Controller	All devices	Reset the Interface	IFC	-
Remote Enable	System Controller	All devices	Enable control of devices via the bus	REN	-
Service Request	Talker	Controller-in-Charge	Inform controller of availability of new information	SRQ	-
End	Talker	Listeners and Controllers	Indicate end of message	END	Last data byte

TABLE 4 UNIVERSAL COMMANDS sent by Controller-in-Charge to all devices.

NAME	*FUNCTION AFFECTED*	*PURPOSE*	*MNEMONIC*	*OCTAL CODE ON DATA LINES*
Local Lockout	Remote/Local	Inhibit Manual Operation of a device locally	LLO	021
Device Clear	Device Clear	Reset instrument to initial state	DCL	024
Serial Poll Enable	Talker	Direct talker to give status next	SPE	030
Serial Poll Disable	Talker	Direct talker to give normal device dependent data next	SPD	031
Unlisten	Listener	Terminate activity as listener	UNL	077

TABLE 5 ADDRESSED COMMANDS sent by Controller-in-Charge

NAME	*RECEIVED BY*	*FUNCTIONS AFFECTED*	*PURPOSE*	*MNEMONIC*	*OCTAL CODE ON DATA LINES*
Go to local	All addressed listeners	Remote/Local	Instruct device to respond to local controls	GTL	001
Selective device clear	All addressed listeners	Device Clear	Reset instrument to initial state	SDC	004
Group Executive Trigger	All addressed listeners	Device Trigger	Start device actions simultaneously	GET	010
Take Control	Controller addressed as talker	Controller	Transfer control to new Controller-in-charge	TCT	011

TABLE 6 ADDRESSES sent by Controller-in-Charge to all acceptors

NAME	*FUNCTIONS AFFECTED*	*PURPOSE*	*MNEMONIC*	*OCTAL CODE ON DATA LINES*
My listen address	Listener	To alert device to next message	MLA	040 to 076
	Talker	To return talker to Idle		
	Remote/Local	To change device to remote control		
My talk address	Talker	To alert device to send next message	MTS	100 to 136
	Listener	To return listener to Idle		
Other talk address	Talker	To return talker to Idle	OTA	100 to 137

Chapter 3

SYSTEM ASSEMBLY

CONTENTS

Para.
1 Physical limitations of a GPIB system
2 System size
3 Connector specification and assignment
5 Cable specification
7 Electrical and timing limitations
7 Options of the electrical specifications
10 Earthing and local power of considerations
12 Signal and message timing
15 Address assignment

Table Page
1 Connector pin assignments ... 2
2 Response and settling times ... 5
3 Codes for address bit patterns ... 7

Fig. Page
1 Typical bus termination ... 3

PHYSICAL LIMITATIONS OF A GPIB SYSTEM

System Size

1. The GPIB has been designed for the interconnection of instruments into systems where distances between devices is short and electrical noise in the environment is relatively low. Typically, it is suited for groups of instruments in a small laboratory or a set of racks. The number of device interfaces on one bus is limited to 15, although it is quite possible to extend the number of instruments beyond this number by using a single interface to service several programmable instruments, particularly if their controllable functions are simple, similar or associated.

2. The total length of cable in a system is also limited to 20 metres or the number of device interfaces times 2 metres, whichever is the less. This cable may be distributed in any way and any pattern of interconnection may be used - linear, branched or star - but if any one length between two interfaces exceeds 4 metres then the maximum data rate may be affected. Cables are normally manufactured with both plug and socket of the bus connector at both ends, enabling them to be stacked piggy-back fashion on the back of an instrument interface, or enabling two cables to be joined to each other.

Connector specification and assignment

3. As mentioned in the introduction, the American Standard and the IEC Standard specify different connectors, but systems with a mixture of both can easily be installed using one adapter cable. The OA 1800 Controller, and all Marconi Instruments GPIB compatible products follow the American Standard.

4. The connector types and pin assignments are shown in the table below, in both cases the female connector is used on all devices, and jack screw facilities are provided to ensure continuous and correct mating.

TABLE 1 CONNECTOR PIN ASSIGNMENTS

	IEEE 488	*IEC 625*
Connector type	*Amphenol series 57 24-way*	*D. type DB255 25-way*
DIO 1	1	1
2	2	2
3	3	3
4	4	4
5	13	14
6	14	15
7	15	16
8	16	17
REN	17	5 with earth on 18
EOI	5	6 with earth on 19
DAV	6 with earth on 18	7 with earth on 20
NRFD	7 with earth on 19	8 with earth on 21
NDAC	8 with earth on 20	9 with earth on 22
IFC	9 with earth on 21	10 with earth on 23
SRQ	10 with earth on 22	11 with earth on 24
ATN	11 with earth on 23	12 with earth on 25
Shield earth	12	13
Logic ground	24	-

Cable specification

5. Any form of cable which conforms to the performance specification below may be used, but the normal construction is for the 3 data management lines and the 5 interface management lines to be wired in twisted pair with separate earths for minimum cross talk. These pairs together with the 8 data lines are contained within an overall shield which is terminated in the connectors at both ends which are themselves shielded.

6. Cable resistance of each signal or twisted pair line:

	0.14 Ω/m max.
resistance of common logic ground:	0.085 Ω/m max.
resistance of shield:	0.0085 Ω/m max.
coverage of shield:	85% min.
capacitance of each signal line to all others connected to ground:	150 pF/m max.

ELECTRICAL AND TIMING LIMITATIONS

Options of the electrical specification

7. The standard specifies the electrical performance of the bus line drivers and receivers in such a way that it is independent of semiconductor technology, but the bus levels do, however, conform to normal TTL levels. The electrical bus interface for each signal line is in three parts, namely resistive termination, bus driver output stage, and bus receiver input stage. The combined stage with the driver off has to present a load impedance to the bus of 2 kΩ with a shunt capacitance of 100 pF max. In order to comply with the normal logical '0' (high voltage) state the resistive termination usually takes the form of 3 kΩ to +5 V and 6.2 kΩ to ground.

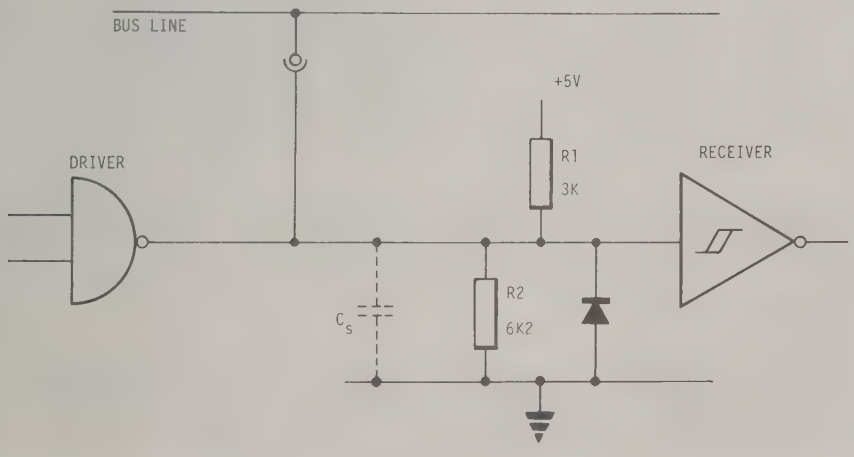


Fig. 1 Typical bus termination

8. All drivers must be able to sink 48 mA to 0.5 V in the logical '1' state, but some signal lines have the option of active pull up to the '0' state for faster bus operation, which is achieved using a three-state type of driver stage. The options on driver types are as follows:

SRQ, NRFD, NDAC:	must always be open-collector type.
IFC, DAV, REN, EOI:	may use open-collector or three-state without restriction.

DIO 1-8:	must use open-collector if parallel poll is implemented, otherwise three-state is permissible.
ATN:	must use three-state if any device in system uses three-state for DAV, EOI or the DIO lines.

Because of its effect on compatibility as well as the speed of data transfer (para. 12) it is important to check the type of driver used in all devices to be connected onto one bus.

9. The standard provides two alternate receiver specifications which may be used in any combination on all bus signal lines. The two types are normal TTL gate and Schmitt trigger with a minimum hysteresis of 0.4 V, of which the latter is preferred because of the improved noise immunity.

Earthing and local power off considerations

10. The voltages for each logical state of the bus lines are defined relative to the logic ground at each devices bus connector. Voltage drops down the bus cable and between device ground for different instruments will reduce the noise immunity of the system by degrading the signal voltages. System cables should therefore be kept as short as possible, particularly if many devices are present, since the cable voltage drops are proportional to bus loading times bus length. The noise immunity will further be reduced if it is intended to run the system with some devices powered off - a procedure which is allowed by the standard provided one more than half the system devices remain on. The effect of devices being powered off is usually to lower the quiescent high state since both terminating resistors in the off device will be to a low potential. This change may be reduced by using at terminating network which includes a diode in series with R1. Even if the high state quiescent voltage is satisfactory, the system user must take care to ensure that the longer settling times required for bus messages are allowed.

11. It is recommended that logic ground is connected to system chassis earth in one point only, to avoid earth loops.

Signal and message timing

12. The table below indicates the response times and settling times required and allowed by the standard. Two of the times are dependent on whether or not three-state drivers are used.

13. The two parameters which mainly determine the rate at which device dependent messages are transmitted are T1, the settling time allowed by the source, and the response time of the slowest acceptor that is addressed. There are no limits to this latter factor, and many instruments slow down the data rate to simplify the operation of the local device by asserting the NDAC line while each byte of information is fully processed, rather than storing it locally. However, if all listeners active during a particular message respond to each byte immediately DAV is received, then with open collector drivers and a maximally loaded bus a data rate of 250 kbytes/s is possible. If three-state drivers are used this rate may be doubled.

TABLE 2 RESPONSE AND SETTLING TIMES

MAXIMUM RESPONSE TIMES DEMANDED BY THE SYSTEM			
IDENTIFIER	*DESCRIPTION*	*FUNCTIONS AFFECTED*	*VALUE*
t_2	Response to ATN change true or false	SH, AH, T, L	200 ns
t_4	Response to IFC true or REN false	T, L, C, RL	100 µs
MINIMUM TIMES FOR PARTICULAR DEVICE FUNCTION STATES			
T_1	Settling time for DIO lines a) Open collector drives b) Three-state drivers on ATN, DIO, DAV, EOI c) As b but second and subsequent bytes	SH	 2000 ns 700 ns 500 ns
T_3	Time to accept interface messages	AH	0
T_7	Controller delay for talker to respond to ATN	C	500 ns
T_8	Time between successive IFC true or REN false	C	100 µs
T_9	Controller delay for EOI receipt a) Open collector driver b) Three state drivers	C	 1500 ns 600 ns

14. To achieve the maximum possible data rate of 1 Mbyte/s additional restrictions must be placed both on the instrument interfaces and on the system, and care must be taken not to reduce the noise margins by allowing insufficient setting time and hence introduce the possibility of message corruption. For maximum data rate:

(1) Use three-state drivers.

(2) Limit the capacitance within each device to 50 pF per signal line.

(3) Use a T1 value of 350 ns.

(4) Use as short as possible lengths of cable between devices, and do not exceed 15 metres.

(5) Add bus terminating resistors equally spaced throughout the bus length to provide a total of 15 unit loads (including the device interfaces).

ADDRESS ASSIGNMENT

15. When a number of devices are connected into a system, listen and talk addresses have to be assigned to each device and the numerical codes representing them written into the software of any controller in the system. In most devices address assignment is achieved by the setting of manual switches on the device interface. The switches correspond to the 5 bits carried on data lines DIO 5 whereas DIO 6 and 7 are fixed and determine whether it is a talk or listen address. It is normal but by no means essential to assign one unique five bit pattern to each device which forms part of both addresses, thus multiple devices can only be addressed together by deliberately programming both address codes one after the other. For example, if two devices were assigned, the address patterns: device A 10110, device B 11010 then their listen addresses are 66_8 ("6") and 72_8 (":") and their talk addresses are 126_8 ("V") and 132_8 ("Z"). To address both to listen at once the controller would send ISO 7 or ASCII characters "6:" over the bus with attention signal true.

16. Two useful variations to the one code per device address assignment are suggested below:

Multiple Address : A device may be assigned two or more usually consecutive addresses thus allowing types of device-dependent message to be differentiated without analysing the syntax of the message. This might be used where a talker can generate two types of message and is particularly cost effective when one device interface is servicing several programmable instruments. This technique reduces bus loading but if done too much can cause problems due to the limited code space.

Duplicated Address : Two listeners but not two talkers may be assigned identical addresses, and this is used when the two devices involved can accept but not necessarily action the same vocabulary of device dependent messages. This most often is possible when one instrument is a functional extension of the other.

17. The following table shows the ISO 7 and ASCII code representation of all the possible address bit patterns in the single byte addressing scheme. In addition to the address switches, the system control, talk only and listen only switches if present must be set as follows:

No Controller System : One talker has talk only ON
All listeners have listen only ON.

Single Controller System : System control ON
All talkers have talk only OFF
All listeners have listen only OFF.

Multi-Controller System : One device only System Control ON
All other switches OFF.

TABLE 3 CODES FOR ADDRESS BIT PATTERNS

DIO LINES					*LISTENER*	*CODE*	*TALKER*	*CODE*
5	*4*	*3*	*2*	*1*	*OCTAL*	*ISO-7*	*OCTAL*	*ISO-7*
0	0	0	0	0	40	Space	100	@
0	0	0	0	1	41	!	101	A
0	0	0	1	0	42	"	102	B
0	0	0	1	1	43	#	103	C
0	0	1	0	0	44	$	104	D
0	0	1	0	1	45	%	105	E
0	0	1	1	0	46	&	106	F
0	0	1	1	1	47	'	107	G
0	1	0	0	0	50	(	110	H
0	1	0	0	1	51	)	111	I
0	1	0	1	0	52	*	112	J
0	1	0	1	1	53	+	113	K
0	1	1	0	0	54	,	114	L
0	1	1	0	1	55	-	115	M
0	1	1	1	0	56	.	116	N
0	1	1	1	1	57	/	117	O
1	0	0	0	0	60	0	120	P
1	0	0	0	1	61	1	121	Q
1	0	0	1	0	62	2	122	R
1	0	0	1	1	63	3	123	S
1	0	1	0	0	64	4	124	T
1	0	1	0	1	65	5	125	U
1	0	1	1	0	66	6	126	V
1	0	1	1	1	67	7	127	W
1	1	0	0	0	70	8	130	X
1	1	0	0	1	71	9	131	Y
1	1	0	1	0	72	:	132	Z
1	1	0	1	1	73	;	133	[
1	1	1	0	0	74	<	134	\
1	1	1	0	1	75	=	135	]
1	1	1	1	0	76	>	136	↑

Chapter 4

MESSAGE FORMAT FOR INSTRUMENTATION

CONTENTS

Para.
1 Basic message format
4 Numeric representation
6 Measurement data
9 Programming or control data
10 Status data
11 Message strings and delimiters

Table Page
1 Message format fields 2

BASIC MESSAGE FORMAT

1. The object of this part of the standard is to define a limited family of preferred message codes and formats in a relatively device independent manner so that the highest degree of compatibility can exist between instruments from different manufacturers and between both simple and complex devices. It should be possible to decode messages with minimum of circuitry and to communicate directly between instruments without the need for translation and re-transmission of messages in alternative formats. The degree of standardisation that is possible in a particular device will, of course, depend to a large extent on the nature of the device and its practical application.

2. The formats described are applicable to three broad types of data. A fourth type, data for display to human operator, by its nature requires a completely free format of character string, normally ISO-7 coded and would only conform to the terminator conventions. The three types considered are:-

(1) Measurement data, i.e. measured results and other data gathered or generated by the instrumentation.

(2) Programmed or Control data, i.e. instrument commands and settings usually generated from the stored test program.

(3) Status data i.e. the contents of the status byte transmitted during serial poll and giving overall information about the condition of a device.

3. The form of message used in types 1 and 2 has three main sections: A header containing codes to indicate the type or meaning of the main data part which follows and the whole message unit is then terminated by special delimiter characters. Within these sections it is useful to define particular

fields not all of which may be present, but which enable consistent decoding of the messages. The table defines the fields and their meaning, contents and preferred character sets.

TABLE 1 MESSAGE FORMAT FIELDS

FIELD	*CONTENTS*	*PREFERRED CHARACTERS*	*USE*
T	Type or quality of data	Letters A → Z	Header
U	Sign of data	+ −	Data value
V	Numeric value of data	Numbers 0 → 9 , .	
W	Exponent notation	E + − 0 → 9	
X	String delimiter	, ;	Terminators
Y	Block delimiter	CR(15_8)LF(12_8)ETB(27_8)	
Z	Record delimiter	ETX (3_8)	

NUMERIC REPRESENTATION

4. There are three types of numeric representation which are Integer, Fixed Point and Exponent Notation. Only the last uses the W field and the U field is optioned for all three.

NR1: Integer Format. Any number of digits 0 to 9 optionally preceded by + or −. Leading spaces are permitted but if fixed length working is used leading zeros are preferred. The point is implied to be after the last digit. e.g. +123 may be coded
123 minimum
or +123 explicit sign
or␣␣+123 leading spaces
or 000123 fixed length
␣ represents space (40_8)

NR2: Fixed Point. As NR1 but with a Single Character for the radix point imbedded in the character string. The full stop or comma may be used for the purpose, but if the latter is chosen, it cannot also be used as a delimiter.
Examples of acceptable representation are:

−1.23 123.0 ␣+0.0123

NR3: Exponent Notation. NR1 or NR2 followed by the W field of letter E meaning "times ten to the power" and a signed integer of maximum two digits.

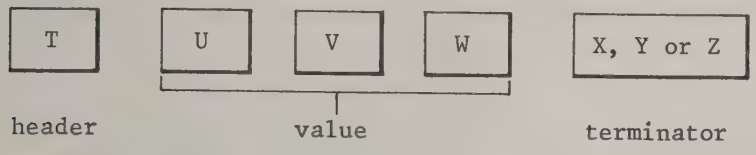
Wherever possible, the exponent should be chosen to convert easily to practical units and hence multiples of three are preferred. To simplify decoding, fixed length exponent is also advised.
e.g. 12300 should be coded 12.3E+03
and may also be coded 12.3E+3

and possibly 1.23E+04
-.000123 should be coded -123E-06
or equally -0.123E-03

5. For non-decimal numbers NR1 may be modified, restricting the range of characters for binary or octal data. Extension of the character range for hexadecimal data is not advised since ambiguity can arise with other data forms.

MEASUREMENT DATA

6. Instrument readings are sent over the bus via the talker function and may typically go to a recording device as well as the controlling device in the System. The general format of the message is chosen to accommodate a wide range of device requirements while still permitting simple decoding. The message can therefore contain the alpha header, any length of numeric values in one of the above forms, and one of the terminator fields.



In common with all messages the fields are transmitted in order left to right and within a field the most significant character is transmitted first.

7. The T field describes the type and quality of the data being sent and may include such information as practical units, parameter code, overflow or overrange flag etc. The standard code for common electrical quantities and multiples are as follows:

Time (Second) :	S	$X10^{9}$	(Giga-) :	G
Current (Amp) :	A	$X10^{6}$	(Mega-) :	MA
Frequency (Hertz) :	HZ	$X10^{3}$	(Kilo-) :	K
Power (Watt) :	W	$X10^{-3}$	(Milli-) :	M
Voltage (Volt) :	V	$X10^{-6}$	(Micro-) :	U
Resistance (Ohm) :	OHM	$X10^{-9}$	(Nano-) :	N
Capacity (Farad) :	F	$X10^{-12}$	(Pico-) :	P
Inductance (Henry) :	H			

Although it is preferred to keep the T field to a minimum number of alpha characters, it is also permitted to pad it out to a fixed length with 'space' codes.

8. The U, V and W fields form the numerical value and follow the rules of notation NR1, 2 or 3. The V field is the only mandatory one.

Choice of terminator will depend on whether the data is by itself or part of a longer message and is discussed in para. 11.

PROGRAMMING OR CONTROL DATA

9. Programming codes are rarely applicable to more than one independent device and standardisation of the code assignment is therefore unnecessary. It is, however, useful for the control information source to have messages constructed in a similar manner and this is usually alternating T and V fields, one pair for each instrument control setting. Providing the V field is always present, delimiters within the series of settings are unnecessary. For example, the setting of a power supply to 5.2 V and 30 mA might be coded.

FIELD	T	V	T	V	T	V	T	X
CONTENTS	V	52	A	30	D	25	S	;
	Voltage Setting		Current Setting		Setting Time		Switch ON	

STATUS DATA

10. Status data is a single byte summary of the overall condition of a device which is sent to the controller as part of a serial poll sequence. In order to cover many different device requirements in a standard way, three bits of the byte are assigned a fixed meaning, and the others may be used in any device dependent way, either being directly assigned 1 bit per function or by coded assignment of up to 32 states. The status byte is as follows:

D107	D106	D105	D108 and D101 - 4
Request Service	Error	Busy	Device dependent bits

Bit 6 indicates if set that the device is in an abnormal or error condition. In such a case the device dependent bits may be coded to give the nature of the error. Bit 5 indicates that the main function of the device is busy (1) or ready (0), i.e. it has completed the programmed task. Bit 7, the request service bit, has already been described in the service request function, but again the device dependent bits may give the reason for the service request in coded form.

MESSAGE STRINGS AND DELIMITERS

11. Basic message units may be joined together to form large arrays of data, and a hierarchical set of terms and delimiters are provided to define and construct them. A string is the smallest group of message units and is transmitted without the talker leaving the active state. A block consists of a number of related strings which are normally generated by the talker with no intervention for additional program data but not necessarily without pauses between strings. The largest data division is a record which includes a number of associated blocks and at the end of which the talker always returns use of the data lines to the controller. The delimiter characters

used for strings, blocks and records are:

	FIELD	*PREFERRED TERMINATOR*
STRING	X	; or ,
BLOCK	Y	ETB or CR and/or LF
RECORD	Z	ETX

Whenever the talker has completed a message the EOI signal may be used instead of a delimiter character on the data lines.

Chapter 5

BASIC LANGUAGE DESCRIPTION

CONTENTS

Para.	
1	Language concept
5	Variables and mathematical capability
8	Editing verbs
9	Delete
10	List
11	REM
12	EDIT
13	Program control verbs
14	LOAD
15	OVERLAY
16	RUN
17	CHAIN
18	END
19	CONT
20	CHANGE
21	BYE
22	Data input and output verbs
23	INPUT
24	ACCEPT
25	PRINT
26	OPEN/CLOSE
27	SCRATCH
28	Program branching verbs
29	GOTO
30	GOSUB
31	GOBUT
32	ONSRQ
33	WAITSRQ
34	RETURN
35	IF... THEN
36	FOR... NEXT
37	Verbs for performing calculations
38	LET
39	DEF
40	DIM
41	Measurement and GPIB communication verbs
42	ASSIGN
43	SEND
44	RECEIVE FROM
45	DELAY

LANGUAGE CONCEPT

1. For several years Marconi Instruments Autotest Systems have been programmed in test languages such as 'INCITE' and 'MI-BASIC' which have been derived from the simple computational language Dartmouth BASIC. With the new system architecture corresponding changes have been introduced into the software to produce a new version of the test language, known as 'XBASIC'. The earlier languages were produced in interpreter form or compiled form, the former having the advantages of instant high level program and instrument manipulation by the operator, and the latter allowing only fast test program execution. 'XBASIC' is constructed in semi-compiler form which retains most of the benefits of both earlier system. This is achieved by packing the source program at load time into an intermediate code which is both fast to execute and which retains a 1:1 relationship to the source program so that it can be automatically regenerated when required for listing or editing etc.

2. The software package contains a BASIC language processor to which may be added any number of modules which:

(1) provide additional commands,

(2) drive specific computer peripherals,

(3) handle the GPIB messages to and from specific instruments.

Thus for a particular set of equipment and range of test capabilities a complete software package may easily be assembled allowing high level test programs to be written and executed. For shorter term applications or when some instrument handler modules are not available, the basic language processor is able to control GPIB instruments in a lower level via general purpose bus commands.

3. BASIC programs consist of a sequential list of statements which have, in general, four parts

LINE NUMBER : COMMAND VERB : QUALIFYING TEXT : TERMINATOR

Line numbers serve as the statement references for both editing and program branching purposes, and when a statement is input without line number it is actioned immediately instead of being added to the stored program. Command verbs are provided for a wide range of operations, some of which are only used in immediate mode (i.e. without line number) for program control or editing. Each command verb will have qualifying text, the syntax of which is chosen to suit the nature of the verb. The terminator may be either the carriage return at the end of line or the full colon between statements on a multi-statement line.

4. The language processor checks the source program syntax at load time and reports any errors. Further error messages can occur while a program is being executed, particularly when fault conditions exist in GPIB instruments or computer peripherals.

VARIABLES AND MATHEMATICAL CAPABILITY

5. The language processor can handle two types of variable, called internal and external variables. The former type are used solely for calculations within the controller and are referred to by a single letter optionally followed by a single numerical digit. External variables are parameters associated with the instrumentation, for example 'frequency range' or 'digital voltmeter reading'. The software module for each instrument will define a name for each programmable parameter so that it may be used in or set by a mathematical expression within the program. The name of an external variable is normally defined in two parts, the name of the instrument and the name of the function separated by ".". Thus the range of the DVM would be altered for example by the expression:

'DVM.RANGE' = B3 + 2

where B3 is an internal variable the current value of which is added to the constant 2 before the new range code is generated and transmitted over the bus to the listener function of the DVM.

6. Internal variables may also be defined in the form of multi-entry tables or multi-dimensional arrays of data. In this case, the variable name is followed by the coordinates of the entry in the array which is being accessed in parameters. For example, the fourth row, second column entry of the array A1 is referred to as A1 (4,2).

7. All mathematical calculation is performed in floating point arithmetic and the results displayed or transmitted where required as a string of ASCII characters. The standard maths package contains routines for addition, subtraction, multiplication, division and exponentation, and optional additional routines are available for logarithmic and trigonometrical functions. Variables may also be defined as string variables, that is the data contained does not represent a single numeric value but a string of usually alphanumeric characters. A full range of string manipulating operators is provided which is useful for constructing composite GPIB messages.

EDITING VERBS

8. Programs are initially generated by simply typing the statement lines in any order into the controller, which automatically stores them in line number order. Typing the same line number a second time with different contents will cause the original entry to be lost and the new one substituted. Similarly, whole lines may be deleted by inputing only the line number and terminator. In addition to these line-by-line operations, the following command verbs may be used to edit and document the test program.

9. DELETE The qualifying text for delete consists of two line numbers separated by a comma, and when executed it results in all existing lines between and including the two specified being expunged from the stored program.

10. LIST This verb is used to display on the v.d.u. screen or print in hard copy lines from any available program. The syntax is

LIST < # channel No. > < file > < first line , last line >

where channel specifies the listing device and file specifies the program name. The qualifying text is all optional and if completely absent LIST will cause the whole of the loaded program to be displayed on the v.d.u.

11. REM REM is short for Remark and is used to append comment for documentation purposes. The statement is ignored at program execution time.

12. EDIT The edit command is followed by a single line number and causes that line to be displayed on the v.d.u. screen. Utilising the cursor controls and substituting and inserting characters the displayed text may be corrected or modified one character at a time and then the whole line is re-combined with the stored program using a terminator key, or when advancing or regressing to edit the adjacent line.

PROGRAM CONTROL VERBS

13. A number of verbs are provided to aid the manipulation of programs, and like EDIT, LIST and DELETE, most do not form part of the stored program but are only used in immediate mode.

14. LOAD The load command is used to transfer a test program from a storage device into the computer. Both the source device, in the form of a channel number, and the program file name are specified following the verb.

15. OVERLAY This command is similar to LOAD except that the specified program is added to that already in computer rather than substituting it. One additional feature of this command is that specified lines only of the new program may be added, allowing common sub-routines, for example, to be transferred without re-typing.

16. RUN Run starts the execution of the test program already loaded at the lowest line number, and the command verb needs no other text.

17. CHAIN If the test program is too long to reside completely in the computer, then it may be divided into several segments which are stored under different file names. The first segment is loaded and run normally and terminates with a CHAIN command which specifies the next segment and start line number in the normal way. When the CHAIN command is reached at run time the next segment is loaded and execution continues automatically.

18. END The end command marks the logical end of the test program and terminates execution. No qualifying text is required.

19. CONT Short for CONTINUE it is a command with no qualifying text which is used to return to program execution after a temporary interruption which was either manual or programmed.

20. CHANGE Change is a general purpose command that allows the operator to alter a number of system parameters. The following are examples of the more common changes that are necessary.

CHANGE TIME 01:45:00	sets the system clock
CHANGE DATE 23-JAN-78	and calender
CHANGE P <channel No>	directs the results printout to a particular device

CHANGE C <channel No>	moves overall system control to a different keyboard console.

21. BYE This simple command returns control to the computer's own operating system - RT11.

DATA INPUT AND OUTPUT VERBS

22. Input and output of data from and to computer peripherals may be achieved during the test program execution by any of the following commands. Data input and output to the instrumentation via the GPIB is achieved by different means, see para. 41.

23. INPUT The input command accepts one or more data value from the specified channel number and assigns it to the single value variables listed in the command. If the list contains external variables then the execution of the command will result in GPIB messages to the appropriate instruments following the receipt of valid data from the correct peripheral channel.

24. ACCEPT The accept command is a direct equivalent of INPUT, but for single dimensional array variables.

25. PRINT The command verb print is used for all output to peripherals whether it results in visible hard copy or screen display or whether it only writes onto a storage medium such as magnetic disk or paper tape. The verb PRINT is followed by the channel number of the destination device and then a string which can consist of variables, expressions to be evaluated before printing or literal text. Formatting characters may also be included.

26. OPEN/CLOSE When the input or output channel number refers to a device using a file structure, for example a magnetic disk unit, then the data will be in block format and all the control data associated with the block header has to be processed. The OPEN and CLOSE commands carry out these housekeeping duties, allowing the input and output commands to be device independent. Thus before a disk file can be read or data recorded in it via PRINT, it must be opened at an appropriate time in the program, and when the last entry has been made the file must be closed.

27. SCRATCH Data files on disk or data entries in computer store may be deleted to make more space available for future computation using the SCRATCH command.

PROGRAM BRANCHING VERBS

28. The normal course of program is through statements in line number order. The following commands are available to alter or branch from that sequence either on interrupt or as a result of programmed data conditions.

29. GOTO GOTO is an unconditional branch to the specified line number.

30. GOSUB This command is a branch to a subroutine which starts on the specified line number. When the subroutine is completed, the control will return to the line following the subroutine call.

31. GOBUT This is similar to GOSUB but it specifies a line number for each control panel button. At that line the servicing routine is written and this is accessed via the GOBUT statement whenever an interrupt is generated by pressing the button in question.

32. ONSRQ This verb, short for GOTO ON SERVICE REQUEST serves the same function as GOBUT, but in a more general way for all GPIB instrument interrupts via SRQ. In this case the qualifying text consists of the assigned device name, followed by the starting line number of the servicing routine.

33. WAITSRQ When program execution has to be suspended until a particular device has completed a task and interrupts via SRQ, then the command WAITSRQ is used together with the assigned device name. When received the request will be serviced via the routine specified in an associated ONSRQ statement.

34. RETURN Return is the terminating statement for a subroutine, or for an interrupt servicing routine, and needs no qualifying text.

35. IF .... THEN A conditional branch is constructed by the command verb IF followed by two expressions to be compared by the relative operator between them. If the condition specified is true, then control passes to the line number following the word THEN instead of the next statement in sequence. The IF statement may also have a single statement after THEN to make it conditional.

36. FOR .... NEXT A section of program may be repeated several times, once for each possible value of a single control variable using this pair of statements. At the beginning of the loop the range and incremended value of the control variable are specified, and the end of the loop is marked by the command NEXT and the name of the control variable. FOR .... NEXT loops may be nested using different control variables for each loop.

VERBS FOR PERFORMING CALCULATIONS

37. Some verbs which evaluate expressions incidentally to their main function have already been described. These include PRINT, IF, FOR .... NEXT. The following commands are provided for computational purposes only.

38. LET LET is the simplest and most general way of assigning to any variable the value of any expression. Both internal and external variables may be set, and floating point or string expressions may be used.

39. DEF This command is used to define a frequently-used expression as a function of a single variable, so that repeated reference to it may be made in an efficient manner throughout the rest of the program.

40. DIM The DIMension command is used to reserve storage space for data tables, arrays, or string variables, and in each case specifies the maximum size of the variable, and must precede the first computational reference to that variable.

MEASUREMENT AND GPIB COMMUNICATION VERBS

41. As explained earlier, the modular concept of the language allows special commands to be created for particular measurement applications, but for general purposes verbs are provided which generate and receive literal GPIB device dependent messages which can then be manipulated or acted upon in the main program.

42. ASSIGN The ASSIGN command is used to specify the name by which a particular device is to be referred throughout the program. After the chosen name the device address and a variable for serial poll response if necessary are quoted.

43. SEND The command SEND can have two parts to the qualifying text. The first is the name of an instrument which has been previously assigned to a particular bus address. This is followed by either an addressed command, a string variable or the literal string to be transmitted to the device. Universal commands can be sent on the bus by simply programming SEND followed by the appropriate mnemonic.

44. RECEIVE FROM The equivalent command for receiving a talker's output message is RECEIVE FROM, and it has similar syntax, except that the second field must be the name of a string variable where the received message will be stored ready for interpretation. The expected terminating character may also be specified.

45. DELAY The last command in the standard language processor is a simple delay for the specified number of milliseconds to allow for test signal or instrument settling times.

Chapter 6

COMMON PROGRAMMING SEQUENCES AND EXAMPLES

CONTENTS

Para.
1 System initialisation from power-on
2 Sending control data and receiving a measurement
3 Serial poll sequence
4 Hand over control of bus from system controller

SYSTEM INITIALISATION FROM POWER-ON

1. In all examples it is assumed that the OA 1800 Controller is the system controller, and it has been assigned the address 00011 which is equivalent to talk address "C" and listen address "#". The initialization is in three parts, namely the clearing of all interfaces, the setting up of remote control, and the setting of devices to a known state.

MESSAGE	*TYPE*	*PURPOSE*
IFC	Control bus signal	Clear interfaces and system controller takes control
ATN	" " "	Force all interfaces to accept multi-line commands
REN	" " "	Enable remote control
LLO	Universal command	Disable local intervention
LAD (1)	Address	Device 1 to remote and listen
LAD (2)	"	Device 2 to remote and listen
LAD (N)	"	Device N to remote and listen
DCL	Universal command	Reset all device functions
or		
SDC	Addressed command	

All of the above steps would be taken automatically on power up by the controller.

SENDING CONTROL DATA AND RECEIVING A MEASUREMENT

2. In this example the controller has only one task and having sent all control data, sets up the bus for receipt of a measurement and waits for the talker to finish the measurement and start transmitting by asserting DAV.

ATN	*MESSAGE*	*TYPE*	*PURPOSE*
T	UNL	Universal command	Clear all listeners
T	MTA 'C'	Address	Address self as talker
T	LAD1 '1'	Address	Address device 1 as listener
F	'R	Device dependent string of control data bytes	Device 1 to range 5
F	5		
F	F		Device 1 to frequency 213
F	2		
F	1'		
F	EOI+3	Final byte and control bus signal	
T	UNL	Universal command	
T	LAD2 '"'	Address	Address device 2 as listener
F	M	Device dependent string of control data bytes	Device 2 to mode 1
F	1		
F	R		Device 2 to range 3
F	3		
F	EOI+E	Final byte and control bus signal	Device 2 execute command
T	TAD2 'B'	Address	Address device 2 to talk will automatically switch off controllers talk function and device 2's listen function provided unlisten on MTA is implemented.
T	MLA '#'	Address	Address controller to listen to measurement result. (Could be achieved by local message).
F	-	Bus signal	Controller releases data bus to device 2
F	1	Measurement data (NR2)	Reading from device 2 is 12.3
F	2		
F	.		
F	3		
F	CR		
F	EOI+LF	Terminator and control bus signal	
T	MTA 'C'	Address	Address self as talker to continue with next control data

SERIAL POLL SEQUENCE

3. In response to a signal on the SRQ signal line (originating in device 2) the controller obtains status data from all appropriate devices in any desired order.

ATN	*SRQ*	*MESSAGE*	*TYPE*	*COMMENT*
T	T	UNL	Universal command	Prevent all other devices listening. Controller listens via local message
T	T	SPE	Universal command	Command talkers to serial poll mode
T	T	TAD1 'A'	Address	Device 1 is addressed
F	T	STB1	Status data	Device 1 responds with status (including DIO-7 RQS false)
T	T	TAD2 'B'	Address	Device 2 is addressed
F	F	STB2	Status data	Device 2 responds with status (including DIO-7 RQS true) device 2 removes its drive of SRQ
T	F	SPD	Universal command	Cancel serial poll mode
F	F	DAB	Device dependent data	Optionally allow device requiring service to talk giving more information than that contained in status

HAND OVER CONTROL OF BUS FROM SYSTEM CONTROLLER

4. Assuming the system controller is also controller-in-charge, it can delegate bus control to any other controller. Only the system controller can demand bus control from the controller-in-charge and this is done in a single step with IFC bus control line Interface clear.

ATN	*MESSAGE*	*TYPE*	*COMMENTS*
T	UNL	Universal command	Unaddress all listeners
T	TAD	Address	Send talk address of next controller-in-charge
T	TCT	Addressed command	Prepares new controller for control transfer
F	-		Control in handed over by releasing ATN
T	Any I/F message		New controller asserts ATN with its first interface message (command or address)

MARCONI INSTRUMENTS LIMITED

SERVICE DIVISION	THE AIRPORT, LUTON, BEDFORDSHIRE LU2 9NS Telephone: (0582) 33866 Telex: 825248
HOME and EXPORT SALES OFFICES	LONGACRES, ST. ALBANS, HERTFORDSHIRE AL4 OJN Telephone: (0727) 59292 Telegrams: Measuretest, St. Albans Telex: 23350
U.S.A.	MARCONI INSTRUMENTS DIVISION, MARCONI ELECTRONICS INC., 100 STONEHURST COURT, NORTHVALE, NEW JERSEY 07647 Telephone: (201) 767-7250 Telex: 710-991-9752
BUNDESREPUBLIK DEUTSCHLAND	MARCONI MESSTECHNIK G.m.b.H., P.O. BOX NO. 210330 JOERGSTRASSE 74 8 MUNICH 21 Telephone: 582041 Telex: 05 24642
FRANCE	MARCONI INSTRUMENTS, 3 AVENUE DU MARECHAL DEVAUX, 91550 PARAY-VIEILLE-POSTE Telephone: 687 36 25 Telex: 203882

A GEC-MARCONI ELECTRONICS COMPANY

www.manualsplus.com

1014534

Printed in England